```c

#include <math.h>
#include "tiff.h"
#include "allocate.h"
#include "randlib.h"
#include "typeutil.h"

void error(char *name);
// initialize limitIntensity function
int limitIntensity(double value);
// initialize applyFilter function
void applyFilter(struct TIFF_img* output_img, struct TIFF_img* input_img, double** usedFilter, int PSF_dim);

int main (int argc, char **argv)
{
    FILE* fp;
    struct TIFF_img input_img, color_img;

    if ( argc != 2 ) error( argv[0] );

    /* open image file */
    if ( ( fp = fopen(argv[1], "rb") ) == NULL) {
        fprintf( stderr, "cannot open file %s\n", argv[1] );
        exit( 1 );
    }

    /* read image */
    if ( read_TIFF( fp, &input_img ) ) {
        fprintf( stderr, "error reading file %s\n", argv[1] );
        exit( 1 );
    }

    /* close image file */
    fclose( fp );

    /* check the type of image data */
    if ( input_img.TIFF_type != 'c' ) {
        fprintf( stderr, "error:  image must be 24-bit color\n" );
        exit( 1 );
    }

    // declare 1D array of double pointers filter
    double* filter[9];

    // iterate through 1D array filter and allocate enough memory for 9 doubles each
    for (int i = 0; i < 9; i++) {
        filter[i] = malloc(sizeof(double) * 9);
```

```
48      }
49      // populate filter array according to h(m,n)
50      for (int i = 0; i < 9; i++) {
51          for (int j = 0; j < 9; j++) {
52              filter[i][j] = 1.0 / 81.0;
53          }
54      }
55
56      /* set up structure for output color image */
57      /* Note that the type is 'c' rather than 'g' */
58      get_TIFF( &color_img, input_img.height, input_img.width, 'c' );
59
60      // declare and initialize integer to store the dimension of the point spread function
61      int PSF_dim = 9;
62
63      // apply filter using applyFilter function as defined below main
64      applyFilter(&color_img, &input_img, filter, PSF_dim);
65
66      /* open color image file */
67      if ( ( fp = fopen( "color.tif" , "wb" ) ) == NULL) {
68          fprintf( stderr, "cannot open file color.tif\n" );
69          exit( 1 );
70      }
71
72      /* write color image */
73      if ( write_TIFF( fp, &color_img ) ) {
74          fprintf( stderr, "error writing TIFF file %s\n", argv[2] );
75          exit( 1 );
76      }
77
78      /* close color image file */
79      fclose( fp );
80
81      /* de-allocate space which was used for the images */
82      free_TIFF( &(input_img) );
83      free_TIFF( &(color_img) );
84
85      return(0);
86  }
87
88  void error(char* name)
89  {
90      printf("usage:  %s  image.tiff \n\n", name);
91      printf("this program reads in a 24-bit color TIFF image.\n");
92      printf("It then horizontally filters the green component, adds noise, \n");
93      printf("and writes out the result as an 8-bit image\n");
94      printf("with the name 'green.tiff'.\n");
```

```
95      printf("It also generates an 8-bit color image,\n");
96      printf("that swaps red and green components from the input image");
97      exit(1);
98  }
99
100 // limitIntensity function definition
101 int limitIntensity(double inputValue) {
102     // declare an integer variable newValue and initialize it to zero
103     int newValue = 0;
104     // if input value parameter is less than zero, assign new value to 0
105     if (inputValue < 0) {
106         newValue = 0;
107     }
108     // if input value parameter is greater than 255, assign new value to
        255
109     else if (inputValue > 255) {
110         newValue = 255;
111     }
112     // otherwise, assign new value to the input value parameter re-cast as
        an integer
113     else {
114         newValue = (int)inputValue;
115     }
116     return newValue;
117 }
118
119 // applyFilter function definition
120 void applyFilter(struct TIFF_img* output_img, struct TIFF_img* input_img,
      double** usedFilter, int PSF_dim) {
121     // declare and define N - the dimension of the point spread function
        (PSF)
122     int N = (PSF_dim - 1) / 2;
123     // declare and define image height and width based on input image TIFF
        struct methods
124     int img_height = input_img->height;
125     int img_width = input_img->width;
126     // declare doubles to store red, green, and blue value for each pixel
127     double redPlane, greenPlane, bluePlane;
128     // declare PSF variables
129     int m, n;
130     // declare variables to store current location within PSF
131     int a, b;
132     // for each pixel:
133     for (int i = 0; i < img_height; i++) {
134         for (int j = 0; j < img_width; j++) {
135             // initialize RGB values to zero
136             redPlane = 0.0;
137             greenPlane = 0.0;
138             bluePlane = 0.0;
```

```
139             // for each pixel in the PSF (9*9 in this case)
140             for (m = -N; m <= N; m++) {
141                 for (n = -N; n <= N; n++) {
142                     // assign a and b to current PSF matrix location
143                     a = i - m;
144                     b = j - n;
145                     // if a and b are within the image boundaries
146                     if (a >= 0 && a < img_height && b >= 0 && b <
                    img_width) {
147                         // apply filter by summing across PSF according to
                     difference equation for 2D filters
148                         redPlane += usedFilter[m + N][n + N] * input_img-
                    >color[0][a][b];
149                         greenPlane += usedFilter[m + N][n + N] *
                    input_img->color[1][a][b];
150                         bluePlane += usedFilter[m + N][n + N] * input_img-
                    >color[2][a][b];
151                     }
152                 }
153             }
154             // populate output image method for color after calling
                  limitIntensity function to ensure acceptable RGB values
155             output_img->color[0][i][j] = limitIntensity(redPlane);
156             output_img->color[1][i][j] = limitIntensity(greenPlane);
157             output_img->color[2][i][j] = limitIntensity(bluePlane);
158         }
159     }
160 }
```